Tracking Changing User Interests through Prior-Leaning of Context

Ivan Koychev

FhG - FIT.ICON
D-53754 Sankt Augustin, Germany
phone: +49 2241 14 2194, fax: +49 2241 14 2146
ivan.koychev@fit.fraunhofer.de

Abstract. The paper presents an algorithm for learning drifting and recurring user interests. The algorithm uses a prior-learning level to find out the current context. After that, searches into past observations for episodes that are relevant to the current context, 'remembers' them and 'forgets' the irrelevant ones. Finally, the algorithm learns only from the selected relevant examples. The experiments conducted with a data set about calendar scheduling recommendations show that the presented algorithm improves significantly the predictive accuracy.

1 Introduction

Recently, many systems have been developed that recommend information, products and other items. These systems try to help users in finding pieces of information or other objects in which the users could be interested [8]. In a similar way, adaptive hypermedia systems build a model of the goals and preferences of each user and use this model to adapt the interaction to the needs of the user [3]. Many of those systems use machine learning methods for learning from observations about the user [15]. However, user interests and preferences can change over time. Some of the systems are provided with mechanisms that are able to track drifting user interests [11, 5, 1, 9, among others]. The problem of learning drifting user interests is relevant to the problem known as concept drift in the area of machine learning. The next section discusses different approaches about learning drifting concept and their applications for learning about users.

In this paper it is assumed that the user interests do not only change, but also possibly recur. The user interests can be quite wide and the user can currently focus her attention on a small subset of her broad interests. For example, the whole set of user interests in the case of Internet browsing can include interests that are relevant to her job, as well as her hobbies, etc. Even the user's job related interests could be quite extensive and interdisciplinary. A system that assists the user in web browsing should be flexible enough to recognize what her current interests are and provide her with relevant recommendations. A possible approach is to learn about current user interests from a time window that includes recent relevant observations only. However, if the

current user interests often change, a precise user profile cannot be learned from a small set of relevant recent observations only. Hence, the system can search for past episodes where the user has demonstrated a similar set of interests and try to learn a more precise description of the current user interests, 'remembering' relevant and 'forgetting' irrelevant observations.

This paper presents such an algorithm for tracking changing user interests and preferences in the presence of changing and recurring context. First, the algorithm learns about current context. Subsequently, it selects past episodes that are relevant to this context and eventually it learns concept descriptions from the selected examples.

The next section discuses different approaches for tracking changes developed in areas of machine learning and user modeling. Section three presents a two-level learning algorithm that is applicable to learning changing and recurring user interests and preferences. Section four presents experiments of the designed algorithm with real data about calendar scheduling preferences as well as with an artificial data set.

2 Related Works

This section briefly introduces different approaches developed for tracking changing (also known as shifting, drifting or evolving) concepts. Such systems use different forgetting mechanisms to cope with this problem. Usually it is assumed that if the concept changes, then the old examples become irrelevant to the current period. The concept descriptions are learned from a set of recent examples called time window. For example, a software assistant for scheduling meetings is described in Mitchell et al. [11]. It employs induction on a decision tree to acquire assumptions about individual habits of arranging meetings. The learning method uses a time window to adapt faster to the changing preferences of the user. A system that learns user's interest profiles by monitoring web and e-mail habits is described in Grabtree and Soltysiak [15]. This research shows that user's interests can be tracked over time by measuring the similarity of interests within a time period.

An improvement of the time window approach is the use of heuristics to adjust the size of the window. Widmer and Kubat [17] use a time window with a flexible size, which is adapted dynamically. The window size and thus the rate of forgetting is supervised and dynamically adjusted by heuristics that monitor the learning process. Klingenberg and Renz [17] investigate the application of such an approach in the area of information retrieval.

Maloof and Michalski [10] have developed a method for selecting training examples for a partial memory learning system. The forgetting mechanism of the method selects extreme examples that lie at the boundaries of concept descriptions and removes from the partial memory examples that are irrelevant or outdated for the learning task. The method uses a time-based function to provide each instance with an age. Examples that are older than a certain age are removed from the partial memory.

Nevertheless, pure time window approaches totally forget the observations that are outside the given window, or older than a certain age. The examples which remain in

the partial memory are equally important for the learning algorithms. This is abrupt and total forgetting of old information which in some cases can be valuable.

System use different approaches to avoid loss of useful knowledge learned from old examples. The CAP system [11] keeps old rules till they are competitive with the new ones. The architecture of FLORA systems [17] assumes that the learner maintains a store of concept descriptions relevant to previous contexts. When the learner suspects a context change, it will examine the potential of previous stored descriptions to provide better classification.

An intelligent agent called NewsDude that is able to adapt to changing user interests is presented in Billsus, and Pazzani [1]. It learns two separate user models: one represents the user's short-term interests and the other represents the user's long-term interests. The short-term model is learned from the most recent observations only. It represents user models that can adjust more rapidly to the user's changing interests. If the short-term model cannot classify the story at all, it is passed on to the long-term model. The purpose of the long-term user model is to model the user's general preferences for news stories and compute predictions for stories that could not be classified by the short-term model. This hybrid user model is flexible enough to consider changes in user interests and keeps track of long-term user interests as well. Chiu and Webb [4] have used a similar approach - a dual student model for handling concept drift.

Webb and Kuzmycz [14] suggest a data aging mechanism that places an initial weight of 1 on each observation. In a similar way Koychev and Schwab [9] have used a gradual forgetting function that provides each observation with a weight according to its appearance over time.

An approach for tracking changing concepts that employs two-level learning algorithms is presented in [16]. The assumption is that the domain provides explicit clues as to the current context (e.g. attributes with characteristic values). A two-level learning algorithm is presented that effectively adjusts to changing contexts by trying to detect (via meta-learning) contextual clues and using this information to focus the learning process. Another two-level learning algorithm assumes that concepts are likely to be stable for some period of time [6]. This approach uses batch learning and contextual clustering to detect stable concepts and to extract hidden context.

The approach presented in this paper also employs a two-learning level. However, it does not assume that the attributes represent current context explicitly. It starts from the assumption that the recent observations are able to provide information about current context. The recent relevant observations cannot be sufficient to learn an accurate description of the concept, but the learned description is accurate enough to be able to distinguish the past episodes that are relevant to the current context. Then the algorithm constructs a new training set, ‘remembers’ relevant and ‘forgets’ irrelevant examples. Finally, the concept description is learned from this set of examples.

3 Tracking Changes through Prior-Leaning of Context

When the concept drifts and possibly recurs, we can use time window based forgetting mechanisms. However, the recent examples that represent the current context can be insufficient for learning accurate descriptions. Therefore, if the context recurs, then remembering the 'old' examples that are relevant to the current context should enlarge the size of the training set and thus improve the predictive accuracy. However, the context is frequently hidden and explicit indicators about its changes and recurrences cannot be discovered easily. Hence, in such cases the aim should be to learn more about the current context and then to search for old observations that were made in a similar context. An algorithm that makes use of this idea consists of the following three steps:

1. *Learning about current context.* A relatively small time window is used to learn a description of the current context (e.g. learning a description of the user interests based on the recent observations about the user).
2. *Remembering relevant past episodes.* The learned description in step 1. is tested against the rest of the training set. The episodes that show a predictive accuracy that is greater than a predefined threshold are selected (i.e. selecting the episodes that are relevant to the current context).
3. *Learning from context-related examples.* The new data set selected in step 2. is used for learning a new description of the current user interests, which is expected to be more accurate.

Let's call this algorithm COPL (COntext Prior Learning algorithm). The COPL algorithm requires a predefinition of the following settings:

- *The size of the time window* used in step 1. This time window should be long enough to allow a sufficiently accurate description of the current context to be learned, as well as short enough to be able to track fast changing user interests. Some enhancements like adaptive time window [17] can be employed aiming at improving predictive accuracy.
- *The episode selection criterion* for step 2. This criterion should be able to distinguish the episodes that are relevant to the learned context in step 1. The criterion should be resistant to noise in the sequence of examples.
- *The threshold for the episode-selecting criterion* in step 2. After the episode selection criterion has been established, a suitable threshold should be defined, which should assure as much as possible that only the relevant old examples be selected.
- *The learning algorithms* used in steps 1. and 3. The same or different learning algorithms can be used in those steps.

Those settings should be defined empirically and based on preliminary investigation of the application domain. The implementation of the algorithm described in the next section gives an example of such definitions.

The next section presents the results from experiments that compare the designed algorithm where the main idea is to extend the set of examples by recovering relevant past examples as opposite to the CAP and FLORA approaches where the model was extended by past rules.

4 Experiments

This section present results from experiments with the COPL algorithm. Two data sets are used in the experiments. The first one contains data from a real use of a calendar manager tool aiming at helping the user to scheduling meetings [Mitchell et al. [11]. The second one is an artificial data set [13] that is used in many papers in the area of Machine Learning dedicated to concept drift (e.g. [17], [10], etc.)

Mitchell et al. [11] have developed a software assistant that helps schedule a particular user's calendar: a calendar manager called CAP (Calendar APprentice). CAP learns the users' scheduling preferences through routine use, enabling it to give customized scheduling advice to each user. It can be considered as an analogy to a human secretary who might assist someone in managing a calendar. CAP employs induction on decision tree to acquire assumptions about individual habits of arranging meetings. The learning method uses a time window to adapt faster to the changing preferences of the user. The newly generated rules are merged with old ones. The rules that perform poorly on the test set drop out of the list.

The user's scheduling preferences depend very much on a hidden context. Some of this context can be assumed and explicitly presented and used for improving predictive accuracy (e.g. academic semesters, etc.). However, there are many other events and conditions that can influence the meeting schedule and which cannot be explicitly represented by an attribute space (e.g. room availability, the schedule preferences of other participants of a meeting and many others). Under this condition, the predictive accuracy of the system can oscillate with very high amplitude. A more comprehensive investigation and analysis of the specifics of the domain can be found in Mitchell et al. [11].

The section below presents the results from experiments conducted with the CAP data set[1]. The attributes used for describing the calendar events in the current experiments are listed in Table 1. The task is to predict the following meeting characteristics:

- *Duration* - the duration of the meeting in minutes e.g. 30, 60, 90, etc. (number of values legal - 13);
- *Day-of-week* - the day of the week of this meeting; e.g. Monday, Thursday, etc. (number of legal values - 6);
- *Location* – the place where the meeting is held; e.g. weh5409 (number of legal values - 142);

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-5/www/cap-data.html

- *Start-time* - the time at which the meeting begins, in military time; e.g. 930 (9:30am), 1400 (2pm), etc.) (number of legal values - 21);

Third-most-common-time-last-60-days-this-meeting-type
Third-most-common-time-last-60-days
Second-most-common-time-last-60-days-this-meeting-type
Second-most-common-time-last-60-days
Most-common-time-these-attendees-last-60-days
Most-common-time-these-attendees
Most-common-time-last-60-days-this-meeting-type
Most-common-time-last-60-days
Most-common-day-these-attendees-last-60-days
Most-common-day-these-attendees
Duration-of-next-meeting-with-these-attendees
Duration-of-last-meeting-with-these-attendees
Day-of-week-of-next-meeting-with-these-attendees
Day-of-week-of-last-meeting-with-these-attendees
Required-seminar-type
Required-course-name
Required-speakers
Single-person?
Action
CMU-attendees?
Group-attendees?
Position-attendees
Department-attendees
Sponsor-attendees
Known-attendees?
Duration
Day-of-week
Location
Start-time

Table 1. The list of features that are used for describing calendar events.

The settings of the algorithm listed in the previous section are defined for the conducted experiments as follows:

- The size of the time window: Preliminary experiments show that for different prediction tasks the size of the window that produces best predictive accuracy can be quite different. For the given data set the best accuracy is reached for the window of the following size: Location - 200; Duration - 350; Start-time - 350; Day-of-week - 400.
- The episode selection criterion for step 2. The criterion used in this implementation selects the examples e_j for the new data set taking into account the average predictive accuracy in its neighborhood. In particular, a small episode around the example which includes the previous two and next two examples, is used. An event will be selected for the new training set $e_j \in S_{new}$ if the average predictive accuracy for this episode is greater than or equal to a predefined threshold τ.
- The threshold for the episode-selecting criterion in step 2. is set up to $\tau = 0.6$ in all experiments.

- The learning algorithm used in steps 1. and 3. is Induction on Decision Tree (aka ID3) [12]. This algorithm was used in CAP, which makes the comparison between different approaches clearer. This algorithm produces an explicit user profile (e.g. set of rules) that is understandable for the user. This is an important advantage from the viewpoint of user modeling.

Prediction task	*CAP*	*ID3-FM*	*COPL (ID3)*
Location	64%	58%	67%
Duration	63%	71%	79%
Start-time	34%	39%	48%
Day-of-week	50%	52%	66%
Average	53%	55%	65%

Table 2. Comparison of predictive accuracy for the User.

Table 2 presents the results from experiments with data for User 1. In this experiment a new description of user preferences is learned after each 10 meetings. The learned description at each step is tested on the next 10 meetings. The line in the table presents the accuracy of prediction for different learning tasks. The results are compared with the CAP. The average predictive accuracy of the ID3 with full memory (ID3-FM) to some extent outperforms the CAP. This is slightly surprising, because CAP is designed to track changing user preferences better than a simple learning algorithm. An explanation of this phenomenon is that some implementation details like attribute selection criteria and used pruning method can change the outcome of the algorithm. The use of one level time window, even with an adaptive size, does not improve the predictive accuracy because the user preferences alternate very often and with high amplitude. The comparison between full-memory learning algorithm (ID3-FM) and the presented two-level learning algorithm is fully compatible because the same implementation of the basic learning algorithm is used. The results from the experiments show that the context-learning algorithm is able to improve the average predictive accuracy for each feature. All those improvements are significant (using t-test with $\alpha = 0.01$).

Figure 1 shows the results from experiments for the predicted features. It can be seen that the user's preferences can change abruptly, which leads to a dramatic decrease of the predictive accuracy. The presented two-level algorithm tracks changes better than the basic algorithm and produces a significantly improved average accuracy.

Experiments with this data set, which use the Winnow and Weighted-Majority algorithms, were reported in Blum [2]. The Winnow with a large feature set reaches the best average accuracy, which is equal to that reached by the algorithm in the presented experiments. However, these algorithms are not suitable for producing explicit user profiles, which is considered to be important in the area of user modeling.

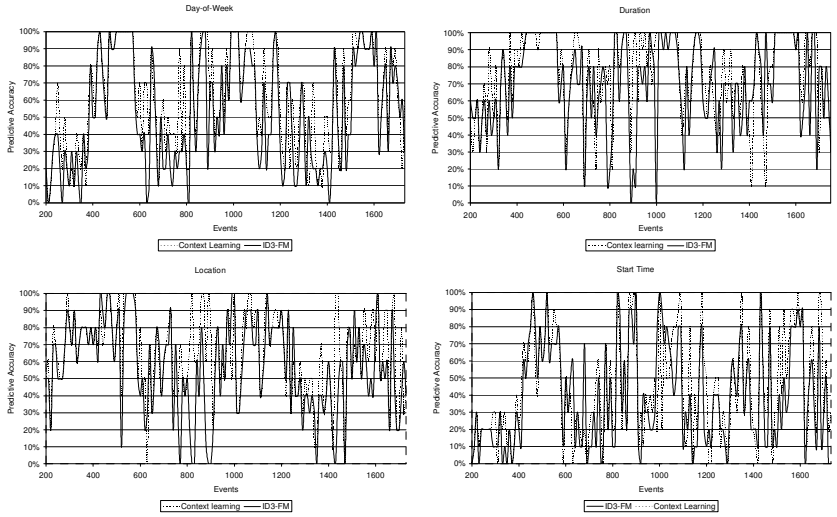

Fig. 1. The improvement in predictive accuracy for the predicted features.

To compare the presented approach with FLORA3, which is able to recover 'old' rules learned in a similar context [17], experiments were conducted also with STAGGER data set [13]. The instance space of a simple blocks world is described by three attributes ***size*** = *{small, medium, large}*, ***color*** = *{red, green, blue},* and ***shape*** = *{square, circular, triangular}*. There is a sequence of three target concepts (1) ***size*** = *small and* ***color*** = *red*, (2) ***color*** = *green or* ***shape*** = *circular* and (3) ***size*** = *(medium or large)*. 120 training instances are generated randomly and classified according to the current concept. The underlying concept is forced to change after every 40 training examples: (1)-(2)-(3). A concept description is learned from initial *n* examples. After each learning phase the predictive accuracy is tested on an independent test set of 100 instances. The result are averaged over 10 runs. The concept recurrence is simulated by generating this sequence three times: (1)-(2)-(3)-(1)-(2)-(3)-(1)-(2)-(3) [17].

The parameters for the COPL algorithm in this experiment are set up as follows: the size of the time window used at step 1 is 18; the episode selection criteria and the related threshold remain the same as above; and the used learning algorithms at step 1 and 3 is Naïve Bayes Classifier (NBC) to demonstrate the ability of the presented two-level algorithm to work with other learning algorithms.

Algorithm\Examples:	*2-120*	*121-240*	*241-360*
FLORA 3 context	85.9%	85.4%	83.5%
COPL (NBC)	85.3%	85.6%	87.1%

Table 3. Comparison between FLORA3 and COPL (NBC) on recurring context.

Table 3 compares the presented algorithm with FLORA3 [17]. On the basic data set (1-120) the FLORA3 produces a slightly better accuracy (i.e. non significant difference). On recurring concepts (i.e. examples 121-360) both algorithms perform better than the ones that do not recover the context (e.g. FLORA2 [17]- 81.5%). The COPL (NBC) algorithm benefits from the recurrence of context better than FLORA3 (see columns *121-240* and *241-360* of Table 3). Moreover, the predictive accuracy of the presented algorithm increases when context recurs, which shows that it really takes advantage of context recurrence. For example, on second recurrence of the concept (see column *241-360* of Table 3) the COPL algorithm produces a significantly better (using t-test with $\alpha = 0.01$) average accuracy than FLORA3.

5 Conclusion

The paper describes a two-level learning algorithm that is able to track changing user interests and preferences through prior-learning of context. The algorithm benefits from the recurrence of the context by remembering the relevant observations and forgetting the irrelevant ones. The presented approach provides a general framework for dealing with changing and recurring user interests that can be used with different machine learning algorithms. Conducted experiments with recommendations about calendar scheduling demonstrate that the approach is able to improve the predictive accuracy significantly. Additional experiments conducted with an artificial data set demonstrate that the presented algorithm really makes use of context recurrence and increases the predictive accuracy when the context recurs. Further investigations of the episode selection criterion and designing a mechanism for its threshold detection are expected to improve the predictive accuracy of the algorithm additionally.

The presented two-level learning algorithm can be embedded in any type of adaptive hypermedia system where some observations during the interaction with the user have been collected and then used to learn about the user. The knowledge learned about the user can then be used to adapt the interaction to the needs of that user. Providing the user with adequate recommendations in the presence of fast changing user's interests and preferences is, for example, vital for many contemporary recommendation systems. Future applications of the algorithm are expected to provide fruitful ideas for the development of mechanism for dynamical adaptation of the algorithm parameters.

References

1. Billsus, D., and Pazzani, M. J.: A Hybrid User Model for News Classification. In Kay J. (ed.), UM99: Proceedings of the Seventh International Conference on User Modeling, Lecture Notes in Computer Science, Springer-Verlag (1999) pp. 99-108.
2. Blum, A.: Empirical Support of Winnow and Weighted-Majority Algorithms: Results on a Calendar Scheduling Domain. Machine Learning 26 (1997): 5-23.
3. Brusikovsky, P. Adaptive Hypermedia. User Modeling and User-Adapted Interaction 11 (2001) 87-110.
4. Chiu, B. and Webb, G.: Using Decision Trees for Agent Modeling: Improving Prediction Performance. User Modeling and User-Adapted Interaction 8 (1/2) (1998) 131-152.
5. Grabtree, I. and Soltysiak, S.: Identifying and Tracking Changing Interests. International Journal of Digital Libraries vol. 2 (1998) 38-53.
6. Harries, M. and Sammut, C. Extracting Hidden Context. Machine Learning 32 (1998) 101-126.
7. Klingenberg, R. and Renz, I.: Adaptive information filtering: learning in the presence of concept drift. AAAI/ICML-98 Workshop on Learning for Text Categorization, TR WS-98-05, Madison, WI, (1998).
8. Kobsa, A., Koenemann, J. and Pohl, W.: Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships. The Knowledge Engineering Review, 16(2) (2001) 111-155.
9. Koychev, I. and Schwab, I.: Adaptation to Drifting User's Intersects - Proceedings ECML2000/MLnet workshop: ML in the New Information Age, Barcelona, Spain, (2000) pp. 39-45.
10. Maloof, M. and Michalski, R.: Selecting examples for partial memory learning. Machine Learning 41 (2000) 27-52.
11. Mitchell, T., Caruana, R., Freitag, D., McDermott, J. and Zabowski, D.: Experience with a Learning Personal Assistant. Communications of the ACM 37(7) (1994) 81-91.
12. Quinlan, R.: Induction of Decision Trees. Machine Learning 1 (1986) 81-106.
13. Schlimmer, J. and Granger, R.: Incremental Learning from Noisy Data. Machine Learning 3, Kluwer Academic Publishers (1986), 317-357.
14. Webb, G. and Kuzmycz, M.: Feature-based modelling: a methodology for producing coherent, consistent, dynamically changing models of agents' competencies. User Modeling and User-Adapted Interaction 5(2) (1996) 117-150.
15. Webb, G. Pazzani, M. and Billsus, D. Machine Learning for user modeling. User Modeling and User-Adaptive Interaction 11 (2001) 19-29.
16. Widmer, G.: Tracking Changes through Meta-Learning. Machine Learning 27 (1997) 256-286.
17. Widmer, G. and Kubat, M.: Learning in the presence of concept drift and hidden contexts: Machine Learning 23 (1996) 69-101.